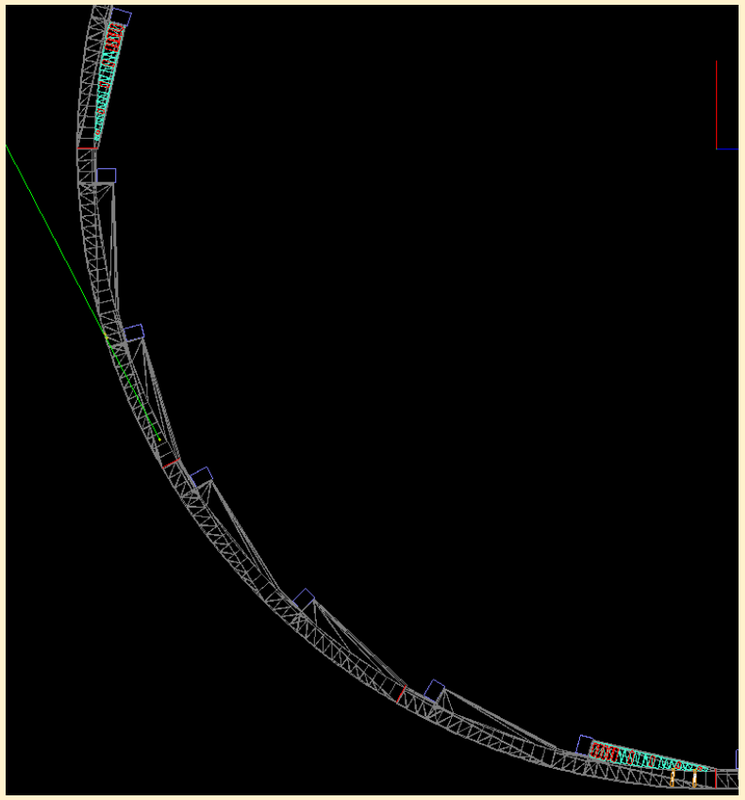
Simulations, Software, & Computing for g-2

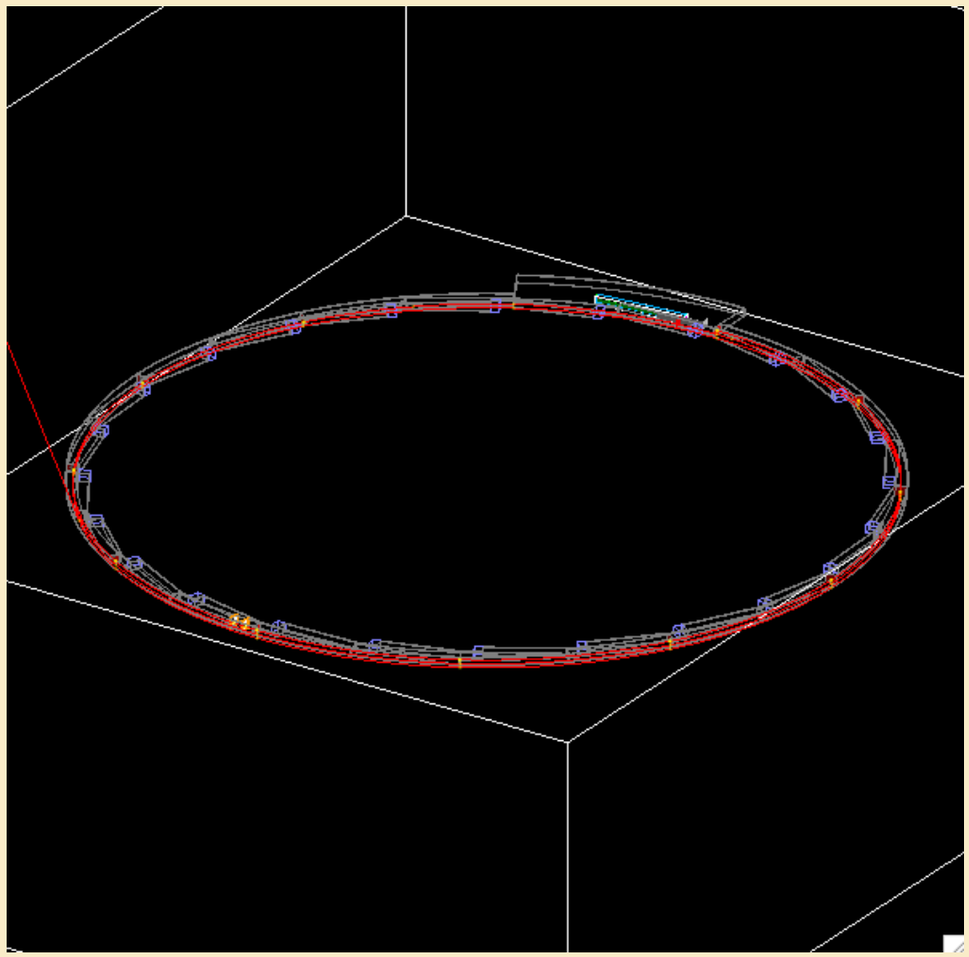


Adam Lyon, Fermilab Scientific Computing Division
December 2012 g-2 Collaboration Meeting

Outline

Computing news and issues
- Bluearc fragility

New tools & services coming to fruition
- JobSub, IFDH, CERNVMFS



Simulation Software – the port to Art
- Workshops
- Art & ArtG4
- g2MIGTRACE to g2ringsim progress

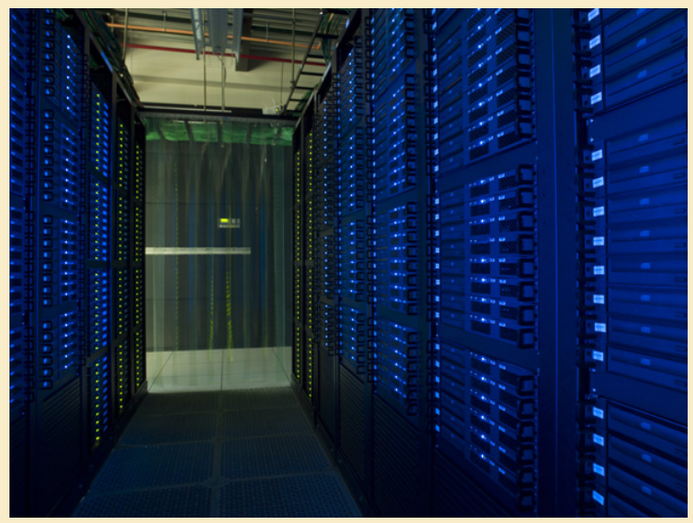
Simulations

Fermilab Computing Resources

3 interactive computing virtual machines (GPCF)

Network Storage "Bluearc"
[High throughput large shared storage]
15 TB now, 50 TB coming
50 TB Cache coming

Fermigrid Batch Slots
25 regular, 25 MARS
200 regular, 50 MARS coming

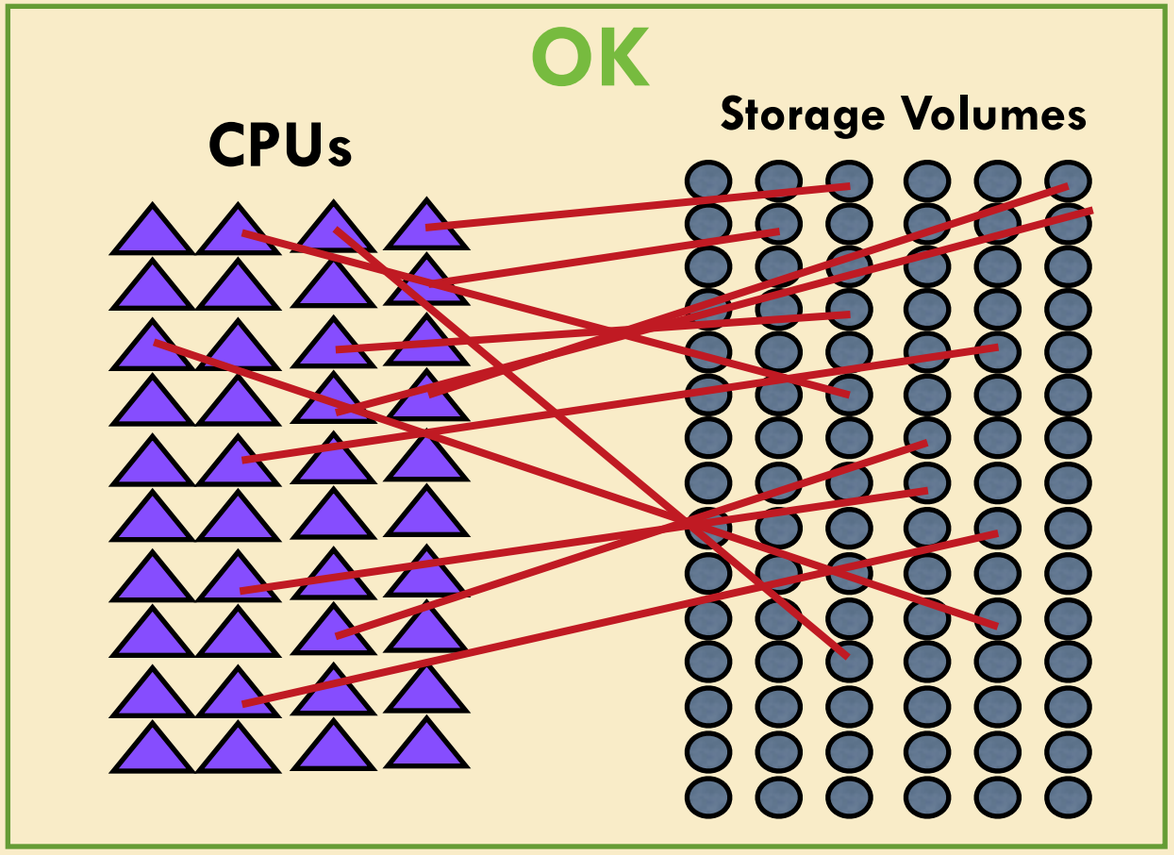


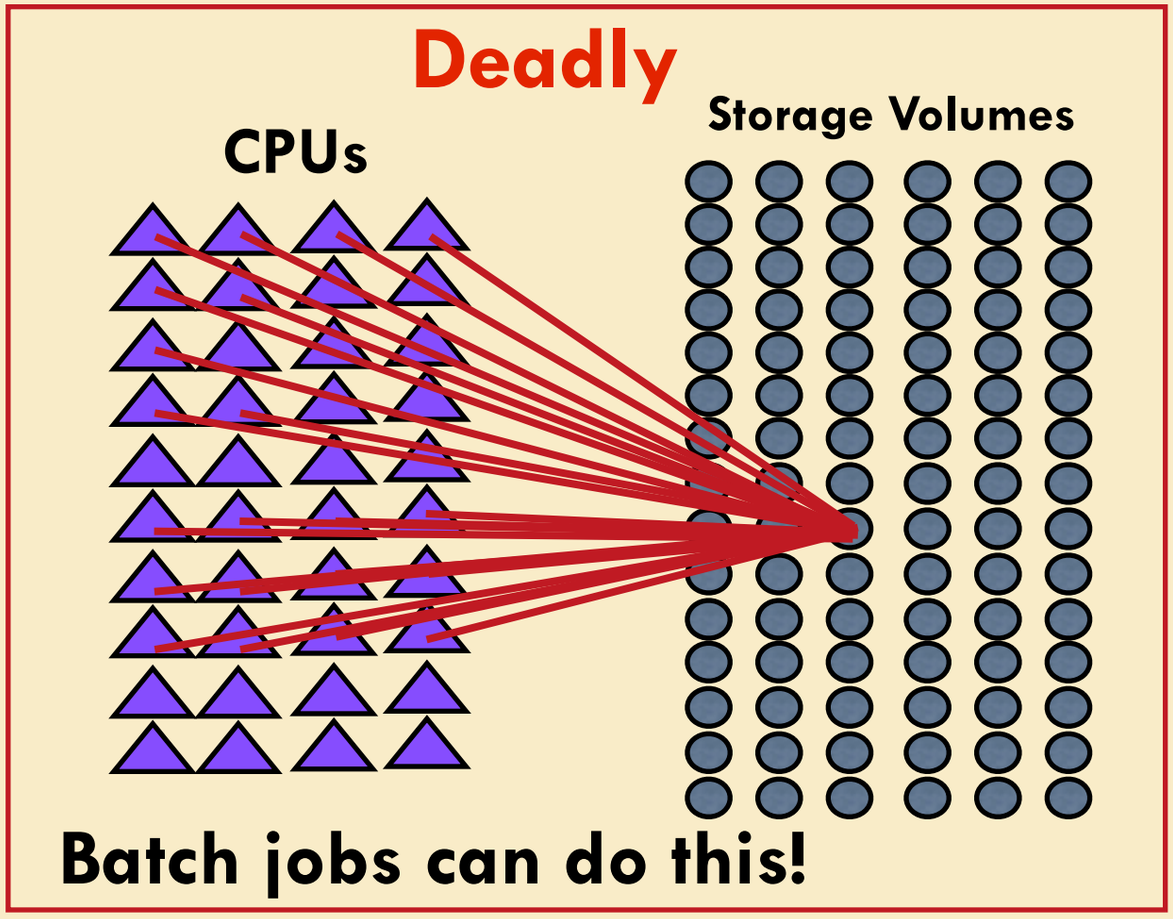
New additions are not confirmed yet

Bluearc crises

Good news:
When used as designed, it works great

Bad news:
When used outside of its design, it kills computing for all of the IF experiments (hard to buy a robust, reasonably priced, 1 PB system)





Bluearc crises

Bluearc volumes (e.g. /gm2/app, /gm2/data) are mounted on Fermigrid. Convenient, but you MUST take steps to protect the disk system.

See
g-2 Wiki (https://cdcvs.fnal.gov/redmine/projects/g-2/wiki) –> Computing & Software –> Running jobs on Fermgrid

Future:
Bluearc mounts on Fermigrid will likely be removed. Replaced with a data handling system

Network storage on gm2gpvm is down

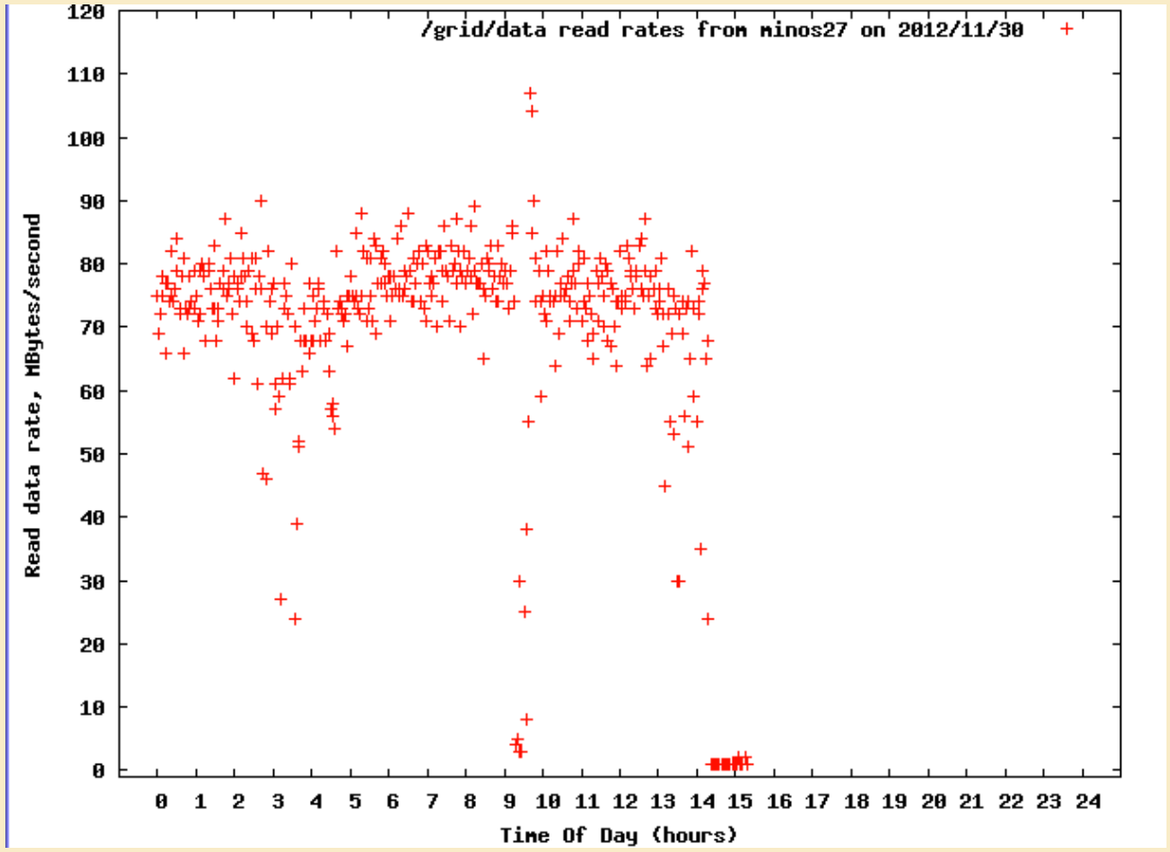
Inbox x

Adam Lyon <lyon@fnal.gov> Nov 30 (13 days ago)
to gm2-detector, Margaret

Hi - We seem to be in the midst of another "Bluearc crisis". /gm2/dat and /gm2/app on gm2gpvm are completely unresponsive.

It is being investigated as I write this. -- Adam

Adam Lyon (lyon@fnal.gov)
Fermi National Accelerator Laboratory
Computing Division / Running Experiments Data Handling Group Leader
D0 & g-2 Experiments
630-840-5522



A suite of Intensity Frontier Computing Tools

Several tools and services becoming available for easier computing

CERNVMFS: Easy installation of software releases on your machines
- **You install the client on your worker nodes, your desktop, your laptop.**
- **You are done!**

- **A release manager updates software on a server somewhere.**
- **You get the updates an hour or two later automagically!!**
  [No more downloading tar files]
- **The system caches files locally. You can see all files, but only download what you use.**
- **Using a prototype server at U Wisconsin.**
- **Testing by NOvA and us. Art on the Mac is distributed using this.**
- **Large scale stress tests very soon**
- **Plans for production system early 2013.**

Suite of tools

JobSub – a standard way to submit grid jobs, to here and remotely (has some fancy features - like "DAGs") [Mostly done - we use it]

SAM – A data handling system for large event and auxiliary files. Takes files from tape, caches, and delivers world wide. Has been integrated with ART Framework. [Will use in the near future]

IFDH – A swiss-army knife of data handling utilities

- **o Handles the "last mile" of file delivery**
- **o Can collect your output files for delivery to Fermilab**
- **o Has Bluearc protection**
- **o Gives access to the Bluearc from remote sites**

[Using for Bluearc protection, other uses soon]

Database Applications: Conditions, Calibration, ACNET capture
[Starting discussions with DB app team soon]

Software with care

Requirements:

- **A software system that allows us to work together easily**
- **A software system that makes our results reproducible**

But I just want to make my plots!

I know - a software system should make "making my plots" easy. And make it easy for you to work with others and for others to work with you.

- **Following best practices**
- **Using standard libraries and APIs**
- **Creating your own libraries for others to use**
- **Share your code in a repository**
- **Documenting your code**

ART Framework - modular

Allows you to write your physics code without worrying about the infrastructure. Makes it easy to work with others. But not for free – you have to learn it.

Some people find such a system constraining:

- **Infrastructure is hidden behind the scenes from you**
- **Your ideas may not be included**
- **You have to trust a system you didn't write**
- **You miss out on the fun of writing super-cool complicated C++ code**

Some people find such a system liberating:

- **You can concentrate on physics code**
- **Your C++ is pretty easy (you are *using* a complicated system, not *writing* it)**
- **You get to miss out having to maintain the complicated system (yay!)**
- **You can use code from others and share yours with others**
- **You can get services for free (e.g. data handling)**

For mainline software, I don't see a choice here – we're relying on ART like NOvA, Mu2e, Microboone, LBNE

Why we need to "evolve" g2MIGTRACE?

- Because it isn't a generic framework
- It's not modular
- It's hard to add new functionality in a "work together easily" way

Case in point...

In g2migtrace/src/primaryConstruction.cc

```
// constructionMaterials is essentially a "materials library" class.
// Passing to to construction functions allows access to all materials

  /**** BEGIN CONSTRUCTION PROCESS  ****/

  // Construct the world volume
  labPTR = lab -> ConstructLab();
  // Construct the "holders" of the actual physical objects
#ifdef TESTBEAM
  ArcH.push_back(labPTR);
#else
  ArcH = arc->ConstructArcs(labPTR);
#endif
  // Build the calorimeters
  //  cal -> ConstructCalorimeters(ArcH);
   station->ConstructStations(ArcH);
#ifndef TESTBEAM
  // Build the physical vacuum chambers and the vacuum itself
  VacH = vC -> ConstructVacChamber(ArcH);
```

In g2migtrace/src/primaryConstruction.cc

```
// constructionMaterials is essentially a "materials library" class.
// Passing to to construction functions allows access to all materials

  /**** BEGIN CONSTRUCTION PROCESS  ****/

  // Construct the world volume
  labPTR = lab -> ConstructLab();
  // Construct the "holders" of the actual physical objects
#ifdef TESTBEAM
  ArcH.push_back(labPTR);
#else
  ArcH = arc->ConstructArcs(labPTR);
#endif
  // Build the calorimeters
  //  cal -> ConstructCalorimeters(ArcH);
   station->ConstructStations(ArcH);
#ifndef TESTBEAM
  // Build the physical vacuum chambers and the vacuum itself
  VacH = vC -> ConstructVacChamber(ArcH);
```

We can’t maintain code like this and our sanity

In g2migtrace/src/primaryConstruction.cc

```
// constructionMaterials is essentially a "materials library" class.
// Passing to to construction functions allows access to all materials

  /**** BEGIN CONSTRUCTION PROCESS  ****/

  // Construct the world volume
  labPTR = lab -> ConstructLab();
  // Construct the "holders" of the actual physical objects
#ifdef TESTBEAM
  ArcH.push_back(labPTR);
#else
  ArcH = arc->ConstructArcs(labPTR);
#endif
  // Build the calorimeters
  //  cal -> ConstructCalorimeters(ArcH);
   station->ConstructStations(ArcH);
#ifndef TESTBEAM
  // Build the physical vacuum chambers and the vacuum itself
  VacH = vC -> ConstructVacChamber(ArcH);
```

WHAT IF WE WANT TO TEST A DIFFERENT DETECTOR?

THIS CODE CRASHES IN OPTIMIZED BUILDS. #IFDEF CREATED A SUBTLE HARD TO FIND BUG

THIS KIND OF CODE IS VERY HARD TO EXCISE LATER

We can't maintain code like this and our sanity

Maintaining sanity is hard

It's not your fault - you just want to do your study

We don't have a system that tries to make things easy

It's not the system's fault - it wasn't written to do that

Writing such a system is hard (need experts)

Learning such a system is non-trivial. That's why we have Simulation Software Workshops (with homework)

October Workshop - ~10 people
December Workshop - +3 new people

What to do with g2MIGTRACE?

There's extremely valuable code in g2MIGTRACE:

- **The complicated geometry of the ring and detectors**
- **The detector response algorithms**
- **The magnetic fields**

Goal: Try to keep that code as unchanged as possible

Instead, reorganize it to fit into Art. The really complicated Geant code is mostly "cut and pasted" into the new system (many files we simply copy)

Steal from others?

<u>What did NOvA do?</u> They have a GDML based simulation; incompatible with g2MIGTRACE

<u>What did Mu2e do?</u> They ported their simulation to ART some time ago. Some useful very routines, but they have icky "uber" code [classes that know about EVERY aspect of the simulation]. e.g. A zillion #includes

```
/ Mu2e include files
#include "GeometryService/inc/GeometryService.hh"
#include "GeometryService/inc/DetectorSystem.hh"
#include "GeometryService/src/DetectorSystemMaker.hh"
#include "GeometryService/inc/WorldG4.hh"
#include "GeometryService/inc/WorldG4Maker.hh"
#include "Mu2eBuildingGeom/inc/BuildingBasics.hh"
#include "Mu2eBuildingGeom/inc/BuildingBasicsMaker.hh"
#include "Mu2eBuildingGeom/inc/Mu2eBuilding.hh"
#include "Mu2eBuildingGeom/inc/Mu2eBuildingMaker.hh"
#include "ProductionTargetGeom/inc/ProductionTarget.hh"
#include "ProductionTargetGeom/inc/ProductionTargetMaker.hh"
#include "ProductionSolenoidGeom/inc/ProductionSolenoid.hh"
#include "ProductionSolenoidGeom/inc/ProductionSolenoidMaker.hh"
#include "ProductionSolenoidGeom/inc/PSEnclosure.hh"
#include "ProductionSolenoidGeom/inc/PSEnclosureMaker.hh"
#include "ProductionSolenoidGeom/inc/PSVacuum.hh"
#include "ProductionSolenoidGeom/inc/PSVacuumMaker.hh"
#include "ProductionSolenoidGeom/inc/PSShield.hh"
#include "ProductionSolenoidGeom/inc/PSShieldMaker.hh"
#include "ProtonBeamDumpGeom/inc/ProtonBeamDump.hh"
#include "ProtonBeamDumpGeom/inc/ProtonBeamDumpMaker.hh"
#include "TargetGeom/inc/Target.hh"
#include "TargetGeom/inc/TargetMaker.hh"
#include "LTrackerGeom/inc/LTracker.hh"
#include "LTrackerGeom/inc/LTrackerMaker.hh"
#include "TTrackerGeom/inc/TTracker.hh"
#include "TTrackerGeom/inc/TTrackerMaker.hh"
#include "ITrackerGeom/inc/ITracker.hh"
#include "ITrackerGeom/inc/ITrackerMaker.hh"
#include "CalorimeterGeom/inc/Calorimeter.hh"
#include "CalorimeterGeom/inc/DiskCalorimeterMaker.hh"
#include "CalorimeterGeom/inc/DiskCalorimeter.hh"
#include "CalorimeterGeom/inc/VaneCalorimeterMaker.hh"
#include "CalorimeterGeom/inc/VaneCalorimeter.hh"
#include "BFieldGeom/inc/BFieldConfig.hh"
#include "BFieldGeom/inc/BFieldConfigMaker.hh"
#include "BFieldGeom/inc/BFieldManager.hh"
#include "BFieldGeom/inc/BFieldManagerMaker.hh"
#include "BeamlineGeom/inc/Beamline.hh"
#include "BeamlineGeom/inc/BeamlineMaker.hh"
#include "GeometryService/inc/VirtualDetector.hh"
#include "GeometryService/inc/VirtualDetectorMaker.hh"
#include "CosmicRayShieldGeom/inc/CosmicRayShield.hh"
#include "CosmicRayShieldGeom/inc/CosmicRayShieldMaker.hh"
#include "ExtinctionMonitorFNAL/Geometry/inc/ExtMonFNALBuilding.hh"
#include "ExtinctionMonitorFNAL/Geometry/inc/ExtMonFNALBuildingMaker.hh"
#include "ExtinctionMonitorFNAL/Geometry/inc/ExtMonFNAL.hh"
#include "ExtinctionMonitorFNAL/Geometry/inc/ExtMonFNAL_Maker.hh"
#include "ExtinctionMonitorUCIGeom/inc/ExtMonUCI.hh"
#include "ExtinctionMonitorUCIGeom/inc/ExtMonUCIMaker.hh"
#include "MECOStyleProtonAbsorberGeom/inc/MECOStyleProtonAbsorber.hh"
#include "MECOStyleProtonAbsorberGeom/inc/MECOStyleProtonAbsorberMaker.hh"
#include "MBSGeom/inc/MBS.hh"
#include "MBSGeom/inc/MBSMaker.hh"
#include "GeometryService/inc/Mu2eEnvelope.hh"
```

ArtG4

Similar to Mu2e's Geant infrastructure (e.g. we copied their Run Manager)

But, ArtG4 is completely detector and action agnostic (no uber code)

You can plug-in/out detectors and actions and configure them with the standard ART configuration file (FHICL) [Geometry compatible with future DB]

Written with Tasha Arvanitis (an exceptional summer student - sophomore at Harvey Mudd College) and approved by ART developers

You write pieces of the simulation (really, override certain functions in a few C++ base classes), ArtG4 puts them together (so you don't make mistakes) and makes Geant go. Simulation "data products" end up in the Art event

Main subject of the software workshops
Homework Q8 - Port Geant example Novice02 - Great learning tool

Using C++2011 – Lots of extremely nice C++ simplifications and language additions that prevent hard to catch mistakes – a great improvement

g2ringsim

Started porting in earnest in November, about 4 weeks ago (note that none of us work on this full time, but we have 2 meetings/week)

Ports completed or nearly completed as of today:

Adam – ArtG4, Lab, Arcs, Vacuum chambers
Brendan K – Magnetic Field, Materials, Inflector, Spin tracking, Event action, Primary Generator, Run action, Stepping action, Tracking/Turn detectors, Geometry service
Peter W – Fiber harp, Collimators, Tracking action
Leah – Stations, Traceback

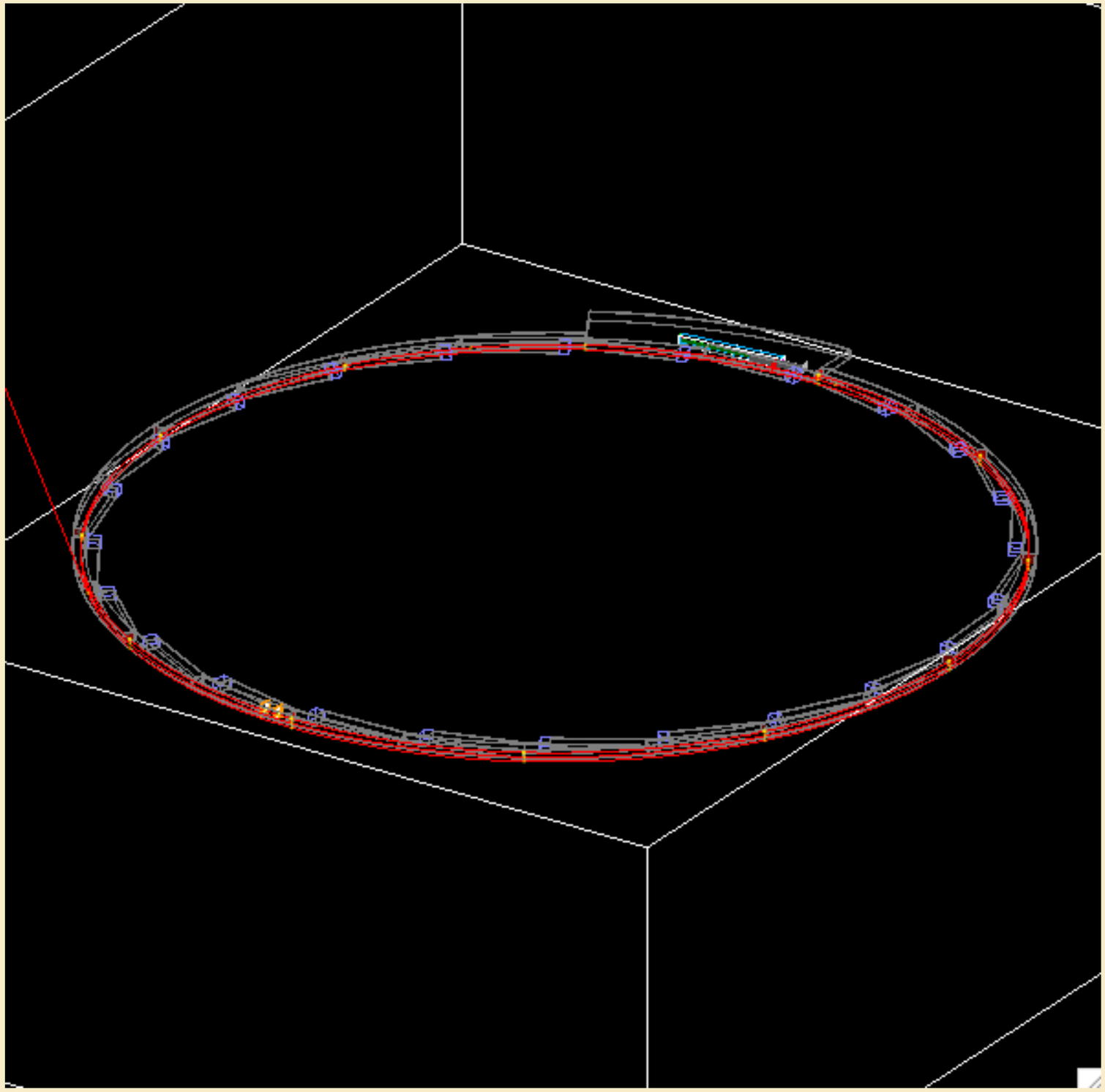
Left to do: Calorimeters & Crystals (Cornell), Straws (Leah), Hodoscopes, Kicker

Need to validate against g2MIGTRACE

BK at 2am: "I figured I'd go to sleep when I hit a show stopper. I haven't hit one yet. I keep coding and it keeps working."

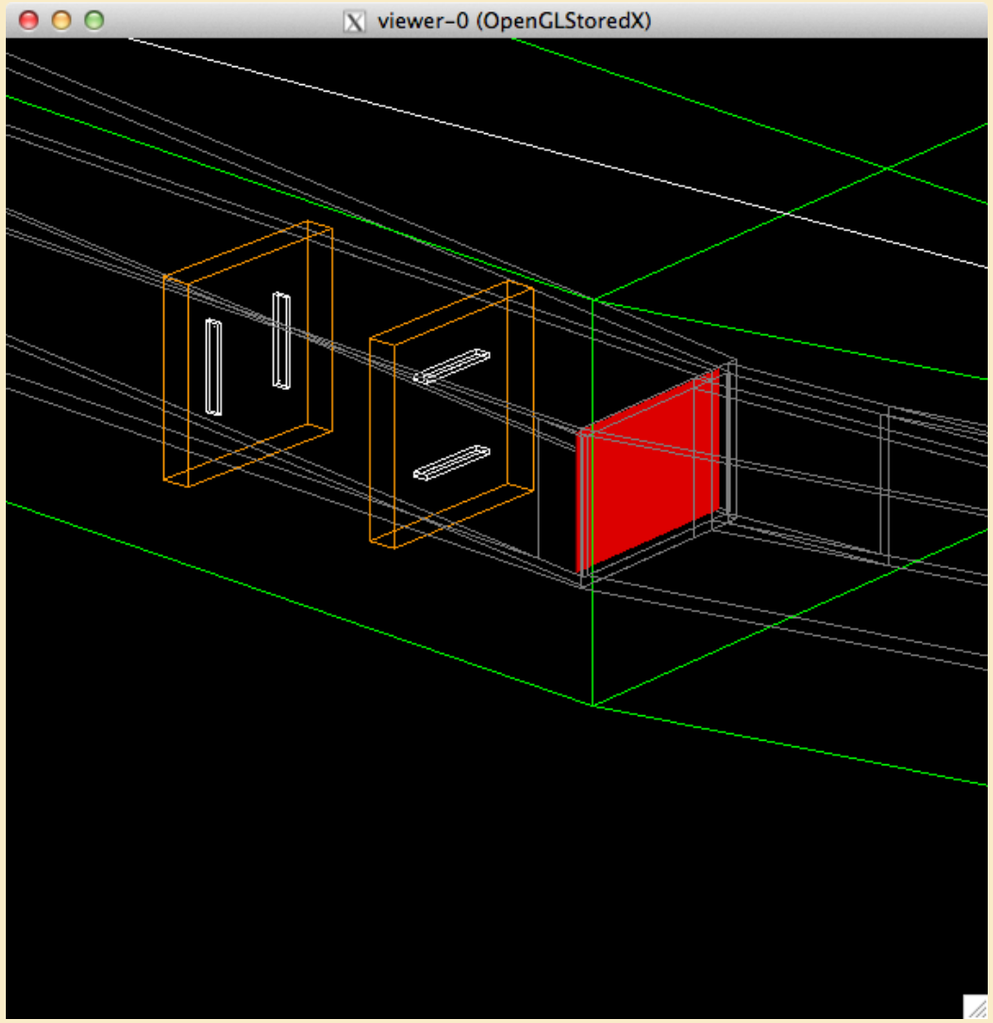
gm2ringsim

Look! Muons go around the ring!

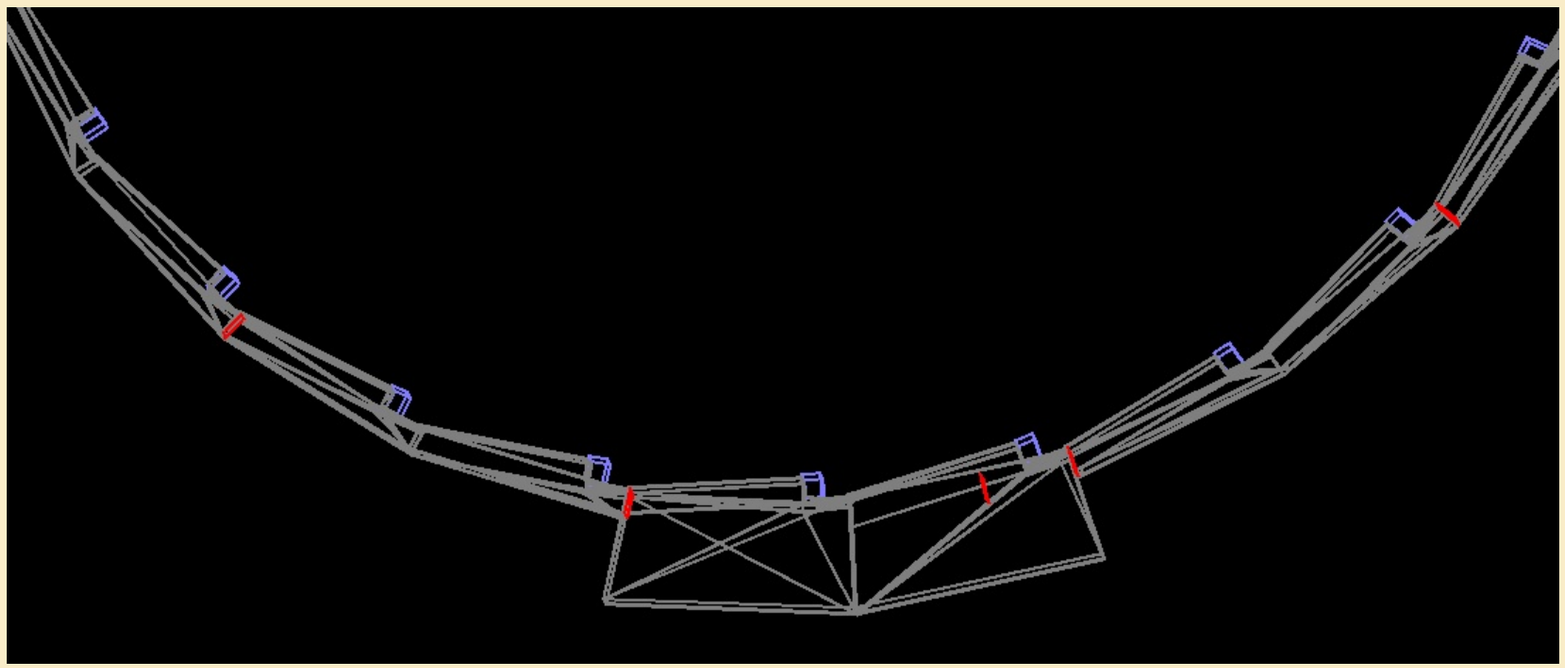


gm2ringsim

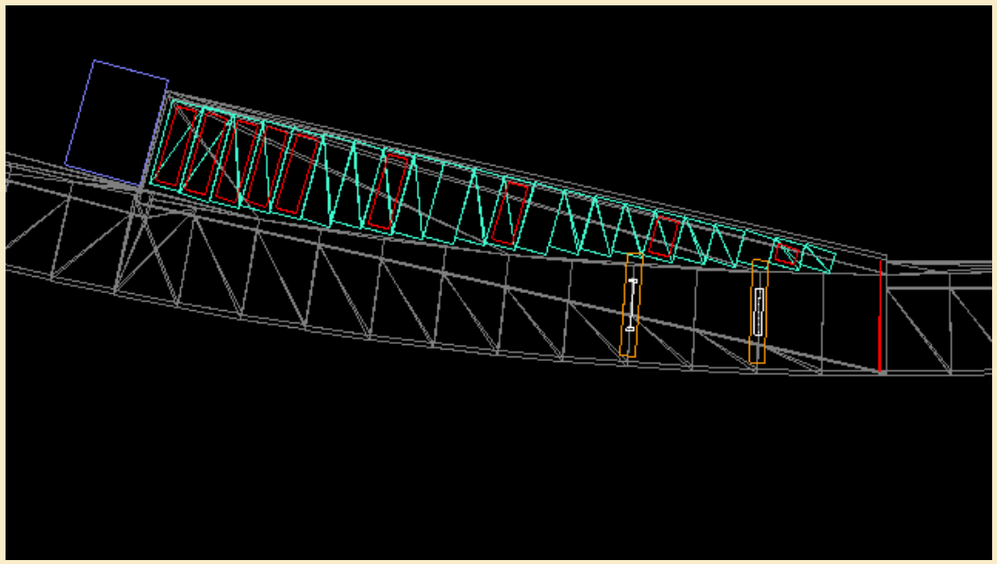
The fiber harp



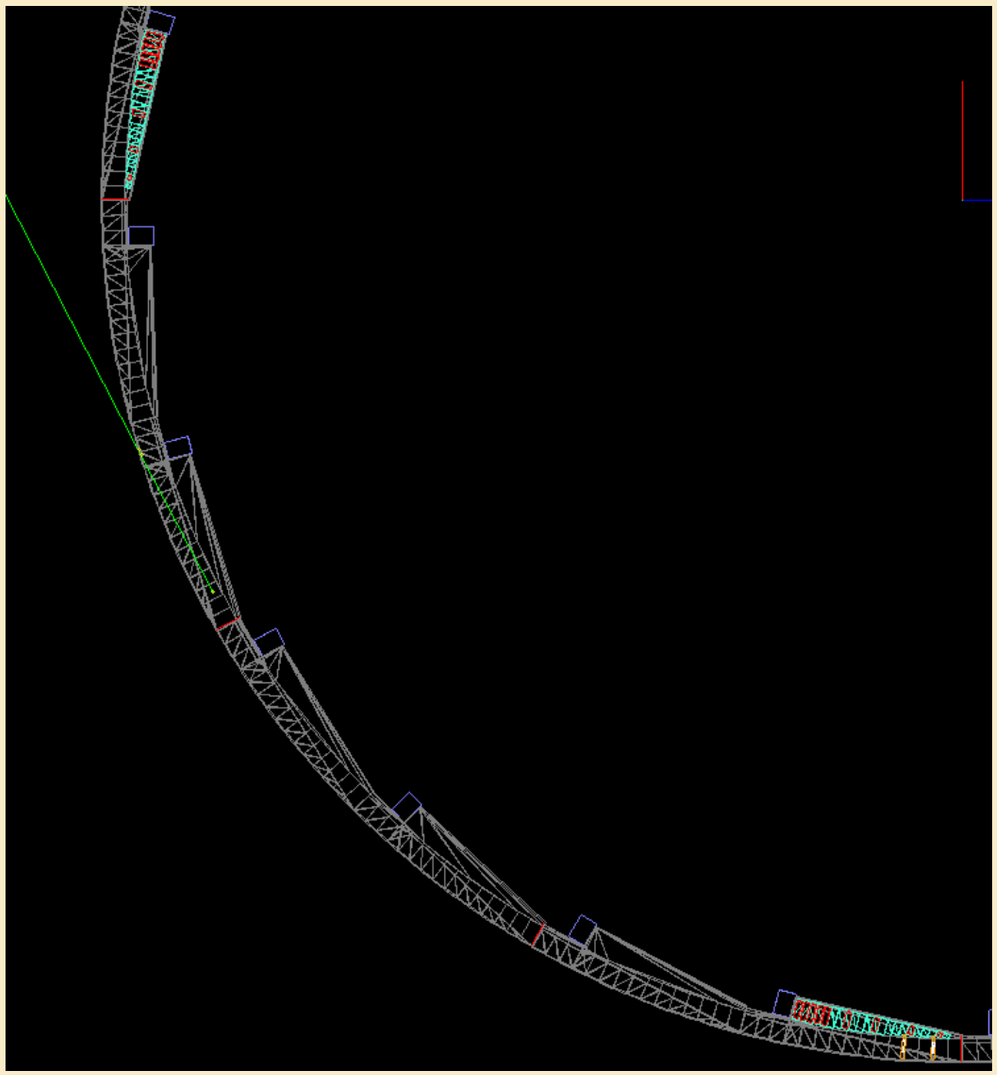
Stations



gm2ringsim

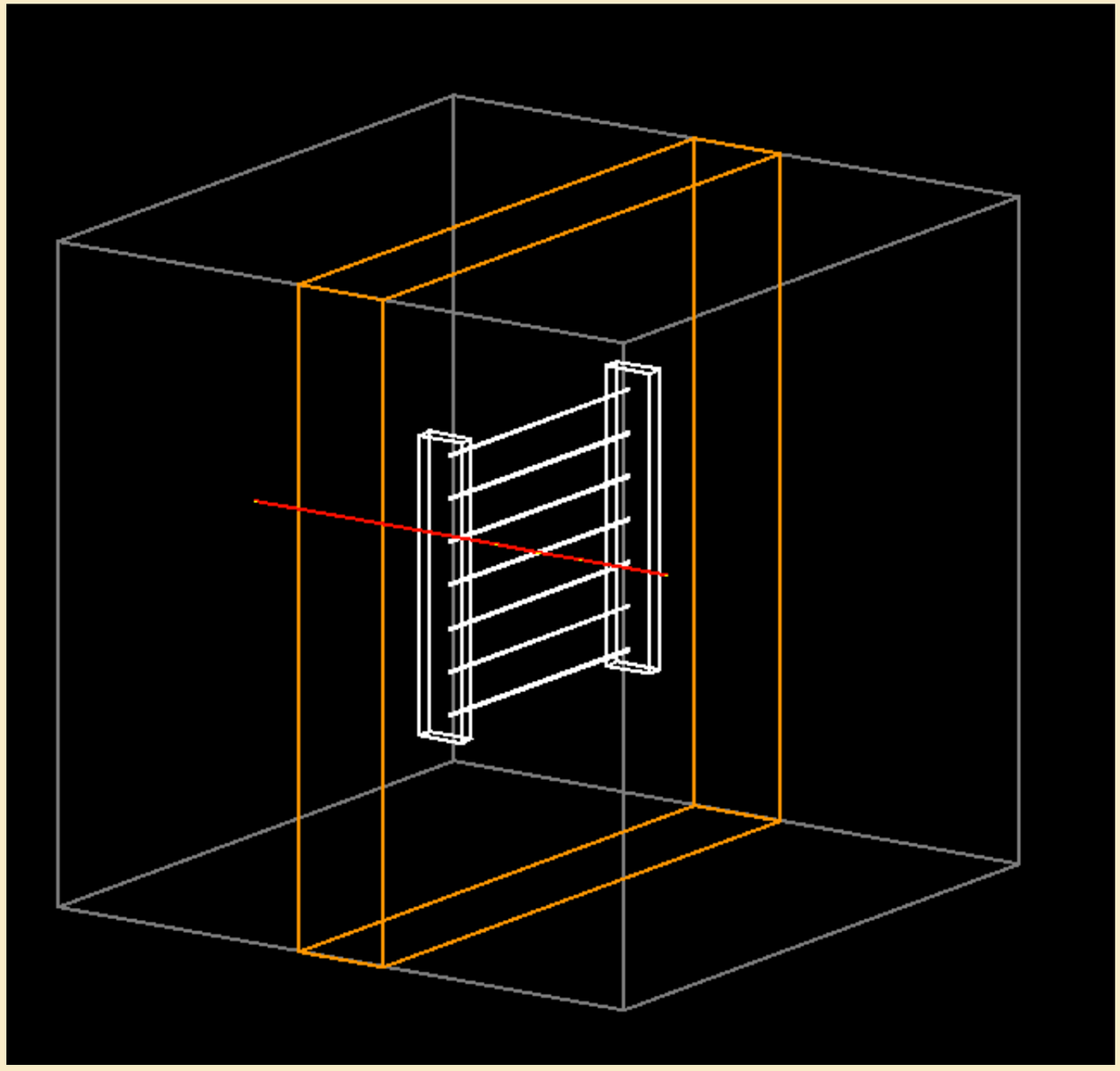


Proto-traceback detectors



A “Test Beam” Simulation

```
user : {

  // Mandatory ArtG4 services
  DetectorHolder: {}
  ActionHolder: {}
  PhysicsListHolder: {}
  RandomNumberGenerator: {}

  // Geometry
  Geometry: {
    world: @local::world_geom
    fiberHarp: @local::fiberHarp_geom
  }

  // Actions
  SimpleParticleSource: {}
  Gm2PhysicsList: {}
  ClockAction: {}

    // Detectors
    World: {}
    FiberHarp: {}

  } //user
} //services

// Override to make a test beam
services.user.Geometry.world.world_x: 100
services.user.Geometry.world.world_y: 100
services.user.Geometry.world.world_z: 100

services.user.FiberHarp.mother_category : world
services.user.Geometry.fiberHarp.nHarps : 1
services.user.Geometry.fiberHarp.RMagicScale : 0
services.user.Geometry.fiberHarp.harpType : [1]
services.user.Geometry.fiberHarp.vacWallPos: [0]
services.user.Geometry.fiberHarp.azimuthalPos : [0]
```



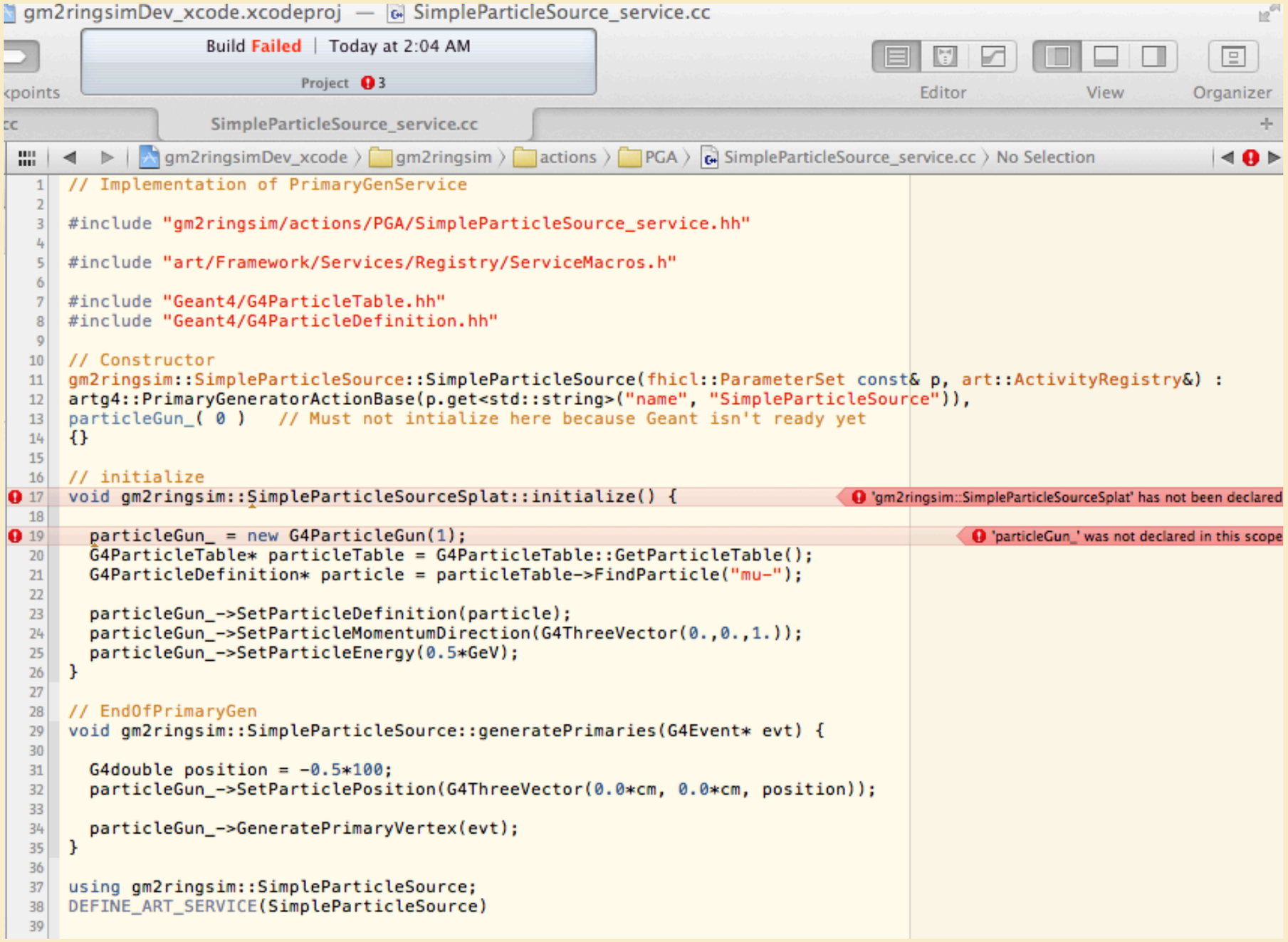
A fiber harp test WITH NO CODE CHANGES (no #ifdefs)

Art, ArtG4, gm2ringsim run on the Mac

Wanted people to have a nice IDE (Apple's free XCode)

Makes coding lots easier!

Distributed to the laptop with CERNVMFS



Last slide

Not going to talk about individual simulations, because everyone has their own talk!!

We're making fast progress with porting g2MIGTRACE to Art and (most importantly) we're making more simulation experts! Expect to be complete in January - we want to use it for our studies!

ART is quite an impressive system. Once you learn it, you can really do things quickly.

Please join us! We'll schedule another workshop soon.

Lots of needed Computing Infrastructure services coming to fruition

We're having fun - Join us!